＜プロジェクト作成～Webサーバー起動＞

Django -admin startproject mysite →mysiteフォルダを作成する

* \_\_init\_\_.py
* Settings.py：ウェブサイトの設定
  + - タイムゾーン
    - 言語コード
    - 静的ファイルのパスを追加
    - ALLOWED\_HOSTを空のリスト→pythonanywhereのホストに変更
    - デフォルトで使うデータベースSQLiteを使用する（セットアップ）
* Urls.py：URLで使用されるパターンのリストが入っている
* Wsgi.py

Manage.py：サイト管理用。これでコンピュータ上でサーバーを起動できる。

1. Settings.pyの設定を書き込む(プロジェクトの作成)
2. Manage.py migrateコマンド→更新ってこと？
3. Manage.py runserverコマンド→Webサーバー起動（<http://127.0.0.1:8000/>）

＜djangoモデルの作成＞

1. ブログを作るためにはブログポストが必要。

・プロパティ：title, text, author, created\_date, published\_date

・メソッド：publish()

1. Djangoモデル（ユーザーやブログポストの情報）はSQLiteデータベースに格納される

・manage.py startapp blogコマンド→blogフォルダを作成する

* \_\_init\_\_.py
* Admin.py
* Apps.py
* Migrations
* Models.py
* Tests.py
* Views.py

1. Djangoで新しいアプリケーションを使えるよう設定の変更をする

・mysite/settings.pyのINSTALLED\_APPSにblogを追加する

1. ブログポストモデルを作成する

・blog/models.pyにclass Post(models.Model): としてモデルを定義（これがオブジェクト）

・↑にauthor, title, text,..を定義していく

・def publish(self):：ブログを公開するメソッド

・メソッドは何かをreturnする→str()を呼ぶとポストのタイトルのテキスト(string)が帰る

1. 作成したモデルをデータベースに追加する

・manage.py makemigration blogコマンド：モデルの変更(新規作成)をdjangoに反映させるためのコマンド

・manage.py migrate blogコマンド：djangoが作成した移行ファイルをデータベースに追加する

＜ログインページと記事作成＞

1. 作成したブログポストを追加・編集・削除するためにdjango adminを使う

・blog/admin.pyを編集する

・admin.site.register(Post)：モデルをadmin(管理)ページで見えるようにするために登録する

・ブラウザでadminページ見れるか確認

・manage.py runserverコマンド→http://127.0.0.1:8000/admin/

→ログインページ！

・manage.py createsuperuserコマンド：管理ユーザーの登録

＜デプロイ＞

1. Gitリポジトリを作成する

・git initコマンド：「.git」フォルダが作成される

・ベースディレクトリ内に「.gitignore」ファイルを作成する（gitで管理しないファイルということ）

・git add –all . コマンド：変更を保存する

・git commit -m “raccoonHP”コマンド：ファイルの変更をリポジトリに記録する

1. Githubにコードをブッシュする

・githubでリポジトリを新規作成

・コマンドGit remote add origin https;//github.com/tanukiyama/raccoonHPgit

Git push -u origin master

1. PythonanywhereでAPIトークンを作成する
2. PythonanywhereでBish→コンソールを起動

・pythonanywhereにWebアプリをデプロイする流れ

コードをGithubからプル→pythonanywhereが認識→デプロイ

・↑を自動でやるツールをインストールする

Pip3.6 install –user pythonanywhere

・githubからアプリを自動的に構成するたえのヘルパーを実行する

Pa\_autoconfigure\_django.py https;//github.com/tanukiyama/raccoonHPgit

・GitHubからコードをダウンロードする  
・PythonAnywhere上にvirtualenvを作成する。  
・一部のデプロイメント設定で設定ファイルを更新する  
・manage.py migrateコマンドを使ってPythonAnywhere上のデータベースをセットアップする  
・静的ファイルの設定  
・APIを通じてPythonAnywhereがあなたのWebアプリケーションを提供するように設定する

・pythonanywhereとPC上のデータベースは別物なのでcreatesuperuserで管理者アカウントを初期化する

Manage.py createsuperuser(混乱をさけるためPC上のと同じにするのがベター)

＜URLの作成＞

1. Mysite/urls.pyを編集して、http;//127.0.0.1:/8000にきたリクエストをblog.urlsへリダイレクトする

From Django.urls import path, includefrom Django.contrib import admin

1. blogディレクトリ内に新たにurls.pyを作成する

・djangoのpathメソッドと、blogアプリのすべてのビューをインポートする

・from Django.urls import path

・from . omport views

・http://127.0.0.1:8000にアクセスがきたらview.post\_listが正しい行先ということをdjangoに伝える

・urlpatterns = [path(‘’, views.post\_list, name=’post\_list’).]

＜中身の作成＞

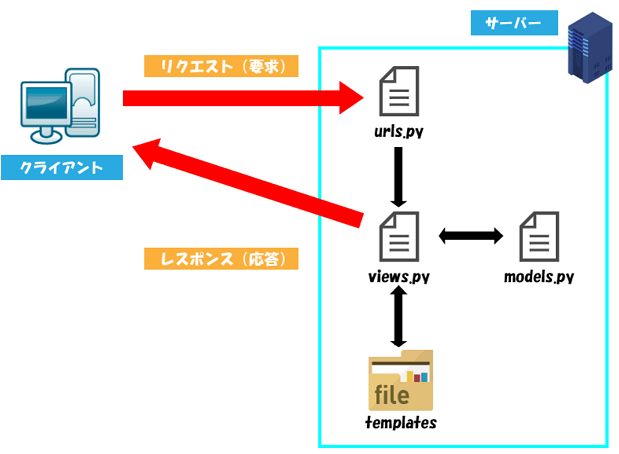
1. blog/views.pyを編集する

・from Django.shortcuts impoer render

・def post\_list(request):

・return render(request, blog/post\_list.html’,{})

・renderメソッドは5~7を行う。データベースのデータを反映させたHTMLページを作り、レスポンスとしてブラウザに返す。

1.ブラウザはサーバーにHTTPリクエストを送信

2. urls.pyから一致するURLパターンを検索

3. 一致したURLパターンに紐づいているviews.py内の関数を実行

4. 関数はモデル（models.py）を通してデータベース操作を行いデータを取得

5. templateフォルダからtemplate\_nameで指定されているテンプレートを取得

6. データベースのデータ（context）をテンプレートに反映

7. 作成されたHTMLをHTTPレスポンスでブラウザに返す

1. テンプレートを作成する

・blogディレクトリ内に”templates“というディレクトリを作り、その中に”blog”ディレクトリを作る。

・この中に“post\_list.html”を作り、コードを書く。

1. デプロイ

・git statusコマンド：更新した概要を教えてくれる

・git add –all .コマンド：gitに対して変更をすべて反映させる（最後の.は現在のディル）→完了

・git commit -m “変更の内容”コマンド：コミットメッセを残す

・git pishコマンド：githubに変更部をアップロード（push）する

・新しいコードをpythonanywhereにpullする

・pythonanywhereのコンソールページを開き、bashに↓を書く

・tanukiyama.pythonanywhere.com

・git pull

・pythonanywhere上の自分のHPを確認する

1. ふい
2. ふい
3. せｒ